Baker 4-12-3-2

A METHOD OF IMPROVING THE LOOKUP PERFORMANCE OF TREE-TYPE KNOWLEDGE BASE SEARCHES

FIELD OF THE INVENTION

This invention relates generally to tree-based searching of a knowledge base, and more specifically, to the use of local memory to improve the speed of the searching process.

BACKGROUND OF THE INVENTION

Many different applications require a search of a large data base of elements, also referred to as a knowledge base, to locate a match with a given search object. In certain applications the search object and the elements of the knowledge base comprise a string of binary bits. An example of such an application is a bridge in a communications system.

A typical communication system includes a number of devices or nodes that communicate over a plurality of connections. The system is organized into a plurality of local connections with a limited number of nodes associated with (connected to) each local connection. A network of bridges interconnects the local connections so that each device can communicate with other devices not associated with the same local connection. The bridge for each local connection monitors input traffic from other bridges in the network to determine if traffic originating at another bridge is addressed to a node on connected to it locally. In response, the bridge provides a path that allows the information to pass through to the local connection. Similarly, when information is sourced from the local connection to an external destination node, the bridge allows the information to pass from the local connection to the next bridge on the path to the destination node.

Typically, the information carried between nodes is in the form of packets of binary bits that travel from the source node to the destination node across the system. A packet typically includes bits identifying the addresses of the packet's source node and the destination node. In one addressing protocol, the address portion of the packet is 48 bits long, with the remainder of the packet comprising payload information bits.

In certain systems, a bridge monitors both internally generated traffic (i.e., traffic sourced at nodes connected directly to the bridge) and also monitors externally-generated traffic (i.e., traffic sourced at nodes external to the bridge) that is broadcast to all bridges of the network. For example, information broadcast over a local area network may not be intended for all network nodes, but is monitored by each network bridge to determine whether any of the intended destination nodes are connected to the bridge. This analysis is performed by maintaining, at each bridge, a knowledge base with an entry for each of the nodes on the bridge's local connection. Thus the bridge receives externally sourced packets and searches its knowledge base to determine whether the 48-bit destination address matches any of the node addresses located on its local connection. The destination address (i.e., the search object) could have a value of any one of 2^48 or about 280 trillion possible addresses. However, the number of entries in the bridge's knowledge base will be equal only to the number of nodes connected locally to it, and therefore will be significantly less than 280 trillion.

Searching a knowledge base to determine a match to a given search object is an important requirement for many different applications. For example, the following applications rely heavily on the performance of speedy searches: data base retrieval; expert systems; robotic and state control strategy; signal recognition, including for example speech and image recognition; communications, including for example data compression and protocol processing for bridging, routing and switching

applications; natural language cognitive systems; modeling operations; parsers; and compilers.

One important attribute of any searching scheme is the worst case time required to complete a search. Generally, searching schemes are implemented in a plurality of steps or cycles that each take a predetermined amount of time to complete. Thus, the maximum time to complete a search is generally reduced by minimizing the time spent at each step of the search.

One popular prior art searching scheme organizes the search space as a tree having a root portion where the search begins, intermediate branches, and finally a plurality of leaves, where the final decisions or matches occur. An exemplary tree structure is illustrated in Figure 1, where entries or decision points are interconnected by branches. An instruction or bit pattern resides at each decision point for analyzing the input bit pattern (also referred to as the search object) and in response thereto for sending the bit pattern to the next appropriate decision point.

The tree of Figure 1 has five levels of analysis or entries, as represented by the five columns of vertically aligned nodes, represented by circles. At a start step 12, a five character word or symbol (the search object) is input to the decision tree, for instance the characters AFGZ3. The most significant characters of the symbol are compared with the tree entries at a decision point 14, and the analysis proceeds along a branch 16, representing the symbol "A," to a decision point 18. From there, the process proceeds as follows: branch 20, decision point 22, branch 24, decision point 26, branch 28, decision point 30 and finally branch 32, which is commonly referred to as a leaf of the tree. At this leaf, the symbols have been decoded and the appropriate action or decision associated with that leaf is executed.

It is known to those skilled in the art that the decision process at each entry of the tree is executed by using a processing device to compare

a first number of symbols at the first entry with a first number of the input symbols. The result of the first comparison determines the next branch that the process will follow. The symbols at the second entry are fetched by the processor and a second group of the input symbols are compared with the symbols at the second entry. These alternating fetching and comparing steps are executed as the search object is processed through the tree until a decision entry is reached.

The decision tree, such as that of Figure 1, is stored in a program memory. Each node of the tree is an instruction and the fields of each instruction specify the branches (links) of the tree to which the processor is directed, based on the results of the instruction. The process then moves along the indicated branch to the next tree node. Special instructions are included at the leaves, as these represent the end of the decision process and therefore command a specific action at each leaf.

With reference to Figure 1 the root of the tree is the first node, node 14. Assume the instruction for node 14 is stored at address 0 in the program memory. Further assume that this instruction maps each letter of the alphabet to a node (i.e., there are twenty-six branches from this instruction or node 14). To find the next node for the process the memory address of the current node (zero in this example) is added to the address offset associated with the matching character. Assume the offset address for the letter A is 10, for B the offset address is 11, for C the offset address is 12, etc. Thus node 18 of Figure 1 is located at memory address 10 (base address of zero plus A offset address of 10). The path for the letter B leads to memory address 11 and the path for letter C leads to memory location 12. Since the example of Figure 1 includes only A, B and C as possible matching values, all other letters of the alphabet (D through Z) that are input to the node 14 are directed to memory locations 13 to 35, each containing a leaf instruction that indicates that the input pattern did not match any location in the tree.

Since the input object begins with an A, the process is directed to the node 18 (memory location 10), which contains three instructions or three potential pattern matches and a memory address offset associated with each. If the pattern match is D and the offset address for the D branch is 1, the process moves to the memory location 11 or node 19 in Figure 1. If pattern match is an E and the offset address for the E is 2, the process moves to memory location 12. If pattern match is an F and the offset address for the F is 3, the process moves to memory location 12, or the node 22 via the link 20. If there is no match at this instruction or node, then the process is directed to an offset address of 4, or node 23, where a leaf with a special instruction is located. For the input symbol presented in Figure 1 (i.e., AF), the process moves to the node 22 (memory location 13) via the link 20.

This process continues for the remaining layers of the tree. Each node of the tree provides an instruction for the processor, and each instruction informs the processor how to interpret the next input bits to determine the address of the next instruction.

A data network classification engine typically utilizes a tree search process to determine various characteristics associated with each data packet or data block that enters the network device, i.e., to classify the input data according to one or more data attributes. Since the data is conventionally presented in the form of binary bits, the classification engine compares groups of the input bits with known bit patterns, represented by entries in the tree structure. A match between the group of input bits and the bits at a tree entry directs the process to the next sequential entry in the tree. The matching processes progress through each entry of the tree until the end is reached, at which point the input bits have been characterized. Because a large number of bits must be classified in a data network, these trees can require many megabits of memory storage capacity.

The classification process finds many uses in a data communications network. The input data packets can be classified based on a priority indicator within the packet, using a tree structure where the decision paths represent the different network priority levels. Once the priority level is determined for each packet, based on a match between the input bits and the tree bits representing the available network priority levels, then the packets can be processed in priority order. As a result, the time sensitive packets (e.g., those carrying video-conference data) are processed before the time insensitive packets (a file transfer protocol (FTP) data transfer). Other packet classifications processes determine the source of the packet (for instance, so that a firewall can block all data from one or more sources), examine the packet protocol to determine which web server can best service the data, or determine network customer billing information. Information required for the reassembly of packets that have been broken up into data blocks for processing through a network processor can also be determined by a classification engine that examines certain fields in the data blocks. Packets can also be classified according to their destination address so that packets can be grouped together according to the next device they will encounter as they traverse the communications medium.

BRIEF SUMMARY OF THE INVENTION

The tree structure for performing the classification process is segregated into a plurality of memory elements, providing the processor with parallel and simultaneous access to the levels of the tree structure. According to the present invention, one or more of the lower level branches of the tree can be stored on-chip with the classification engine, (i.e., the processor) thereby reducing the read cycle time for the lower level tree entries. Advantageously, there are fewer lower level tree entries as these

appear near the tree root. Therefore, the on-chip storage requirements are considerably less than the storage requirements for the entire tree.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention can be more easily understood and the further advantages and uses thereof more readily apparent, when considered in view of the description of the invention and the following figures in which:

Figure 1 is a prior art tree structure for processing an input symbol;

Figures 2 through 5 are block diagrams of the processor and memory elements according to various embodiments of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

According to the teachings of the present invention, the tree structure is partitioned between one or more memory elements, such that depending on the memory elements chosen (i.e., faster memory on-chip versus slower off-chip memory) different read access times are available and thus certain tree entries, that is nodes or instructions as discussed above, are accessible faster than others.

As shown in Figure 2, a processor 50 communicates bidirectionally with memories 52 and 54, where the instructions representing one or more tree levels are stored. For example, tree levels one (referred to as the root level), two and three of Figure 1 are stored in the memory 52 and tree levels four and five are stored in the memory 54. If the memory 52 has a faster memory access time than the memory 54, then the instructions stored in memory 52 can be accessed faster than those in memory 54. It is know that a significant number of tree searches are terminated in the root memory or within one or two levels of the root memory. Simulation and analysis show that about 30% of the tree instructions are matched in the

root tree memory. Thus if the tree root is stored in the memory 52, the process will likely converge faster.

The use of two separate memory structures is merely exemplary as additional memory structures can also be employed for storing levels of the tree. Selection of the optimum number of memory elements, the memory access time requirements of each, and the tree levels stored in each memory element can be based on the probability that certain patterns will appear in the incoming data stream. The tree levels or sections of tree levels that are followed by the most probable data patterns are stored in the memory having the fastest access time. For example, all the input patterns traverse the lower levels of the tree, thus these lower levels can be stored within a memory having a fast read cycle time to speed up the tree analysis process.

The teachings of the present invention can also be applied to parallel processing of tree structures. See Figure 3 where processors 60 and 61 each have a local memory 62 and 63, respectively, for storing lower level tree branches, and a shared remote memory 64 for storing higher level tree branches. Each processor 60 and 61 includes an execution engine, each having an execution pipeline and a program memory. Thus, according to the embodiment of Figure 3, each processor 60 and 61 can search its respective local memory for executing the tree search then access the shared remote memory 64 when the higher tree branches are encountered. In one embodiment, the local memories 62 and 63 are located on the same integrated circuit device as their respective processors 60 and 61, thus providing faster access times than the remote memory 64.

In the embodiment of Figure 4, a processor 70 communicates with a local root memory 72 and a processing engine 74 communicates with an remote non-root memory 76. When a tree search references an instruction that is in the other engine's memory, the latter processor is given control over the searching process to execute that instruction. To provide faster

memory access the local memory 72 is located on-chip with the processor 70.

In another embodiment, a search engine processor is multi-threaded, allowing it to execute a plurality of simultaneous searches throughout one or more search trees. For example, the processor 50 of Figure 2 can fetch the tree structure information from the memories 52 and 54 in parallel, since each memory is accessible through a different processor thread, thereby reducing the time required to execute the classification process.

In another embodiment, as illustrated in Figure 5, an internal memory 80 is included on the same chip as a processor 82, i.e., on-chip. An external memory 84 is located off-chip. In this embodiment, the lowest or root levels (for example, two levels) of the tree are stored in the internal memory 80, referred to as root tree memory. Since there are fewer tree branches at the root level, the capacity requirements for the internal memory 80 are lower than the capacity requirements for an equivalent number of upper level branches. The latter are stored in the external memory 84, which can run at a slower speed (resulting in a higher data latency). But this latency factor has less impact on the speed at which the tree analysis process is executed because these higher tree branches are not traversed as frequently. The use of internal memory and external memory allows each to be accessed in parallel by the pipelined processor 82. Also, use of the internal memory reduces the pin-out count of the integrated circuit incorporating the processor 82 and eliminates signal speed reductions due to impedance mismatches at the pin interfaces. In another exemplary embodiment the tree structure is stored in three memory elements, two memory elements external to the processor and the third on-chip.

It has been shown that the storage of the lower tree branches on-chip reduces the number of clock cycles required to traverse through an

average size tree from about 30 to 40 clock cycles according to the prior art, to about two or three clock cycles according to the teachings of the present invention. Depending on the structure of the particular tree, many of the search processes may terminate successfully at a lower level branch in the on-chip memory, and thereby avoid traversing the upper level branches stored in the slower memory.

In yet another embodiment, it may be possible to store especially critical or frequently–used small trees entirely within the internal memory element 60. Thus providing especially rapid tree searches for any tree that is located entirely on-chip. The segregation between the tree levels stored within the internal memory 60 and the external memory 64 can also be made on the basis of the probabilities of certain patterns in the input data.

Typically, the data input to a network processor using a tree characterization process is characterized according to several different attributes. There will therefore be a corresponding number of trees through which segments of the data packet or data block are processed to perform the characterization function. According to the present invention, the lower level branches are stored on-chip and the higher-level branches are stored off-chip. To perform the multiple characterizations, a pipelined processor will access a lower branch of a tree stored in the on-chip memory and then move to the off-chip memory as the tree analysis progresses. But since the off-chip access time is longer, while waiting to complete the read cycle off-chip, the processor can begin to characterize another aspect of the input data by accessing the lower branches of another on-chip tree. In this way, several simultaneous tree analyses can be performed by the processor, taking advantage of the faster on-chip access speeds while waiting for a response from a slower off-chip memory.

In another embodiment, certain portions of the tree (not necessarily an entire tree level) are stored within different memory elements. For

example, the most frequently traversed paths can be stored in a fast on-chip or local memory and the less-frequently traversed paths stored in a slower remote or external memory.

The tree according to the present invention is also adaptable to changing system configurations. Assume that the tree is processing a plurality of TCP/IP addresses. When the process begins the tree is empty and therefore all of the input addresses default to the same output address. The tree process begins at the root and immediately proceeds to the default output address at the single leaf. Then an intermediate instruction or decision node is added to direct certain input addresses to a first output address and all others to the default address. As more output addresses are added, the tree becomes deeper, i.e., having more branches or decision nodes. According to the teachings of the present invention, the growth of the tree can occur in both the local and the remote memory elements.